Part 2: Assembling clearing files #2

Open **robertying-stripe** opened this issue on Nov 16, 2023 · 0 comments

robertying-stripe commented on Nov 16, 2023 • edited

> # Read the *entire* issue before you start; the docs are linked inline!

Let's take that payment intent data and do something useful with it!

One of the key steps in processing a payment is taking a payment intent – when a user charges a credit card, for example – and assembling those into clearing files. Clearing files are files that contain all the payment details from successful payment intents, and instructs our financial partner to pay Stripe (so we can then send money to our user).

In the real world, we often SFTP this file to our financial partners. To simulate this behavior, we've provided a service that you can POST your file to.

Part 2 (a)

Write a program which takes as an input a file containing payment intents, and prints that file out in the clearing file format. The clearing service expects to receive files in a specific format - here's an example corresponding to the first payment intents file.

You can test the correct formatting of this file by using the web UI. It will return the string `success` and HTTP status code 200 if it accepts the file.

Part 2 (b)

Modify your program to send payment intents files (in the clearing flie format) programmatically. You're welcome to either pass the list of files in (e.g. as command-line arguments) or use a constant holding the files. Files uploaded under the same API key are considered part of the same session.

When You're Finished

Move on to #3

Information about the clearing service

A brief summary of the endpoints important for these tasks:

- `/` : GET from this endpoint to see a hosted page that can be used to call the other APIs for debugging.
- `/create_api_key` : POST to this endpoint to create an API key that you can use for future interactions with this service. The API key should be provided to all the following APIs as a bearer token, i.e. a header `Authorization: Bearer <token>` .
- `/submit_clearing_file` : POST to this endpoint with your clearing file in the `clearing_file_contents` field.
- `/reset` : POST to this endpoint to clear all data

Please note that the service does have rate-limits - a few requests per second is fine. If you get rate-limited, wait a bit and it will cool back down.

It can be accessed at https://reconciliation-uw5gplbdim.stripesandbox.com/.

Full documentation is available here

HTTP Library recommendations

If you're not familiar with the HTTP protocol, this Wikipedia article is a good reference and this article explains a little about HTTP methods and JSON.

Making HTTP requests can be no fun! We recommend these libraries for your language of choice. (This is not an exhaustive list! Ask your interviewer if your language of choice is not on here!)

If there is an HTTP client library that you prefer,
feel free to use that instead.

Language	HTTP Library
C++	cpr
C#	HttpClient
Clojure	clj-http.client
Common Lisp	drakma
Go	net/http
Java / Kotlin	OkHttp
JavaScript (Node.js)	node-fetch or axios
PHP	curl or guzzle
Python	requests
Ruby	excon or httprb
Rust	reqwest or reqwest-blocking
Scala	sttp or http4s
Swift	url-loading

Known Issues

Importing node-fetch in JavaScript

Starting with v3, `node-fetch` no longer supports the Node `require()` syntax and requires users to use ESM modules and the `import` syntax. This requires users to either rename their file or pass additional configuration to Node: https://stackoverflow.com/questions/69041454/error-require-of-es-modules-is-not-supported-when-importing-node-fetch.

For the purpose of this interview, we recommend installing v2 of `node-fetch` instead, using `npm install node-fetch@2` .

Hints

- If you go to the clearing service, you'll see a basic HTML UI that has a bit more information about the file format(s) and lets you make requests.
- An ARN is an acquirer reference **number**. You get to choose the value, and it uniquely identifies a given payment intent that you've sent to the clearing service.
- Feel free to use the current system time (in milliseconds) for the timestamp
- If you want to fully automate your script, you can create a new API key on each run!